대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2003-0016209
Application Number

출 원 년 월 일 : 2003년 03월 14일
Date of Application MAR 14, 2003

출 원 인 : 주식회사 안철수연구소 외 1명
Applicant(s) Ahnlab, Inc., et al.

2003 년 04 월 29 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0006
【제출일자】	2003.03.14
【국제특허분류】	H04L
【발명의 명칭】	제어흐름과 자료흐름을 고려한 악성 행위 패턴 감지 방법
【발명의 영문명칭】	METHOD TO DETECT MALICIOUS CODE PATTERNS WITH DUE REGARD TO CONTROL AND DATA FLOW
【출원인】	
【명칭】	주식회사 안철수연구소
【출원인코드】	3-1999-902882-2
【출원인】	
【성명】	홍만표
【출원인코드】	4-1999-008549-0
【대리인】	
【성명】	박창남
【대리인코드】	9-2001-000437-2
【포괄위임등록번호】	2003-015642-9
【포괄위임등록번호】	2003-016061-0
【대리인】	
【성명】	진천웅
【대리인코드】	9-1998-000533-6
【포괄위임등록번호】	2003-015641-1
【포괄위임등록번호】	2003-016062-7
【발명자】	
【성명의 국문표기】	이성욱
【성명의 영문표기】	LEE,Sung Wook
【주민등록번호】	690408-1018827
【우편번호】	440-320
【주소】	경기도 수원시 장안구 율전동 샘내마을삼호아파트 211-906
【국적】	KR

【발명자】

【성명】	홍만표
【출원인코드】	4-1999-008549-0

【발명자】

【성명의 국문표기】	조시행
【성명의 영문표기】	CHO,Si Haeng
【주민등록번호】	620219-1029514
【우편번호】	140-040
【주소】	서울특별시 용산구 산천동 192 리버힐삼성아파트 109-1102
【국적】	KR

【심사청구】 청구

【취지】 특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인
박창남 (인) 대리인
진천웅 (인)

【수수료】

【기본출원료】	15	면	29,000 원
【가산출원료】	0	면	0 원
【우선권주장료】	0	건	0 원
【심사청구료】	2	항	173,000 원

【합계】 202,000 원

【감면사유】 중소기업

【감면후 수수료】 101,000 원

【첨부서류】 1. 요약서·명세서(도면)_1통 2.중소기업기본법시행령 제2조에의한 중소기업에 해당함을 증명하는 서류_1통

【요약서】

【요약】

본 발명은 제어흐름과 자료흐름을 고려한 악성 행위 패턴 감지 방법에 관한 것이다.

이러한 본 발명은 검사 대상인 두 문장에 각각 포함된 토큰(변수 또는 상수)값들의 실행 중 일치 여부를 판단하여 악성 코드 패턴을 감지하되, 두 문장의 토큰이 모두 상수일 경우와, 두 문장의 토큰 중 하나는 상수이고 하나는 변수인 경우, 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지는 경우, 및 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지지 않는 경우로 구분하여 토큰값들의 실행 중 일치 여부를 판단하는 것을 특징으로 한다.

본 발명에 따르면, 종래의 변수명 비교에서 발생할 수 있는 긍정 오류를 배제하고 부정 오류를 낮추어 악성 행위 감지의 정확도를 향상시킬 수 있다.

【대표도】

도 4

【색인어】

악성 코드, 제어흐름, 자료흐름, 스크립트, 악성 코드, 패턴, 감지

【명세서】

【발명의 명칭】

제어흐름과 자료흐름을 고려한 악성 행위 패턴 감지 방법{METHOD TO DETECT MALICIOUS CODE PATTERNS WITH DUE REGARD TO CONTROL AND DATA FLOW}

【도면의 간단한 설명】

도 1 은 정적 분석에 의한 악성 스크립트 감지 방법에 대한 개념을 설명하기 위한 메일을 통해 자기 복제를 수행하는 비주얼 베이직 스크립트 코드의 실예,

도 2 는 정적 분석에 의한 악성 스크립트 감지 방법에서 나타나는 긍정 오류의 실예,

도 3 은 정적 분석에 의한 악성 스크립트 감지 방법에서 나타나는 부정 오류의 실예,

도 4 는 본 발명에 따라 악성 행위 패턴을 감지하기 위한 검사 대상 문장의 구분개념도,

도 5 는 본 발명에 따른 제어 흐름 그래프의 실예,

도 6 은 본 발명에 따라 악성 행위 패턴을 감지하기 위한 흐름도이다.

*도면의 주요부분에 대한 부호의 설명

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<8> 본 발명은 악성 스크립트에 존재하는 악성 행위 패턴을 감지하는 방법에 관한 것으로서, 특히 정적 분석을 이용하여 악성 행위 패턴을 감지하되 제어흐름과 자료흐름을 고려한 감지 방법에 관한 것이다.

<9> 악성 스크립트의 감지에는 이진 코드를 위한 기법들을 그대로 이용하거나, 소스 프로그램 형태인 스크립트에 적합하도록 다소간의 변형을 가하여 적용하는 것이 일반적이다. 특히, 스캐닝(scanning)을 통한 시그너쳐 인지(signature recognition)는 가장 보편적으로 사용되고 있는 악성 코드 감지 방식이다. 이 방식은 하나의 악성 코드에만 존재하는 특별한 문자열들을 탐색함으로써 해당 코드의 악성 여부를 진단하기 때문에 진단 속도가 빠르고 악성 코드의 종류를 명확하게 구분할 수 있다는 장점을 가지고 있다. 그러나, 알려지지 않은 악성 코드에 대해서는 전혀 대응할 수 없는 문제점이 있다.

<10> 한편, 알려지지 않은 악성 스크립트에 대한 감지 기법 중에서 가장 현실적인 대안으로 받아들여지고 있는 것은 정적 휴리스틱 분석 기법이다. 이 기법은 악성 행위에 자주 이용되는 코드 조각들을 데이터베이스화 하여두고 대상 코드를 스캔하여 그 존재 여부나 출현 빈도를 탐색하여 악성 코드를 감지하는 방식이다. 이 방식은 속도가 비교적 빠르고 높은 감지율을 보이나, 긍정 오류가 다소 높다는 단점을 가지고 있다. 따라서, 이러한 단점을 보완하기 위해서 정적 분석을 이용한 악성 스크립트 감지 방법이 제안되

었다. 이 방법은 악성 행위를 구성하는 메소드 시퀀스의 존재 뿐 만 아니라 관련된 파라미터와 리턴값들까지 검사하기 때문에 단순한 정적 휴리스틱 분석보다는 상당히 정확한 감지 결과를 보이게 된다.

<11> 도 1 은 정적 분석에 의한 악성 스크립트 감지 방법에 대한 개념을 설명하기 위한 악성 비주얼 베이직 스크립트 코드의 실예이다. 도 1 에서 확인할 수 있는 것처럼 다수의 메소드 호출이 하나의 악성 행위를 구성하기 위해서는 반드시 그것들의 파라미터와 리턴값 사이에 특별한 관계가 존재하여야 한다. 예컨대, 4행의 Copy 메소드는 현재 실행중인 스크립트를 'LOVE-LETTER-FOR-YOU.TXT.VBS' 라는 이름으로 복사하고, 7행의 'Attachments.Add' 메소드는 그 파일을 새로 만들어진 메일 객체에 첨부함으로써 메일을 통한 자기 복제를 달성한다. 그러나, 메소드 호출의 존재유무만을 검사하는 방식을 사용하게 되면, A라는 이름으로 스크립트 파일을 생성하고 B라는 이름의 파일을 첨부하는 관계없는 메소드 호출이 존재하여도 이를 악성 코드로 간주하기 때문에 높은 긍정 오류를 보이게 된다. 이에 반해 정적 분석에 의한 감지 방법은 메소드 호출의 존재 뿐 만 아니라, 사용된 파일명, 'fso', 'c', 'out', 'male' 등과 같이 모든 관계있는 값들이 일치하는가를 검사함으로써, 단순한 문자열 탐색보다 정확한 감지 결과를 얻을 수가 있다.

<12> 그러나, 정적 분석 기법 또한 감지 정확도 있어서는 여전히 문제점을 가지고 있다. 종래의 정적 분석 기법은 단지 외형상 드러난 변수명 만을 비교하므로, 주어진 두 개의 변수가 같은 이름을 가지고 있다는 것만으로 이 두 변수의 값이 실행 중에도 같을 것으로 간주하는 오류를 범하게 된다. 도 2 는 정적 분석 기법에서 긍정 오류가 발생될 수 있는 예이다. 종래의 정적 분석 기법은 1행과 4행에서 사용된

변수 'c'가 외형상 동일한 변수명인 것만을 확인하고 두 값이 같은 것으로 간주하게 된다. 그러나, 프로그램을 분석해보면 3행에서 변수 'c'의 값이 새롭게 정의되었으므로 실제 실행시에는 1행의 'c'와 4행의 'c'가 서로 다른 값을 가지게 됨을 알 수 있다. 도 3은 도 2 와는 반대의 경우로서, 정적 분석 기법에서 부정 오류가 발생될 수 있는 예이다. 종래의 정적 분석은 1행의 'c'와 3행의 'd'가 서로 다른 변수이므로 두 값이 일치하지 않는다고 판정하나, 실제 실행시에는 2행의 복사문인 "d=c"에 의해 두 변수의 값이 같아지게 된다. 결론적으로, 상술한 두 가지 형태의 오류는 전체 악성 행위 패턴 감지의 입장에서 볼 때, 각각 긍정 오류와 부정 오류를 유발하기 때문에 이러한 오류를 해결할 수 있는 방법이 필요하다.

【발명이 이루고자 하는 기술적 과제】

<13> 이에 본 발명은 상기와 같은 필요성에 부응하기 위해 안출된 것으로서, 정적 분석시 제어흐름과 자료흐름을 고려하여 정확도를 향상시킬 수 있는 악성 행위 패턴 감지 방법을 제공하는데 그 목적이 있다.

<14> 상기와 같은 목적을 달성하기 위하여 본 발명에 따른 제어흐름과 자료흐름을 고려한 악성 행위 패턴 감지 방법은 검사 대상인 두 문장에 각각 포함된 토큰(변수 또는 상수)값들의 실행 중 일치 여부를 판단하여 악성 코드 패턴을 감지하되, 두 문장의 토큰이 모두 상수일 경우와, 두 문장의 토큰 중 하나는 상수이고 하나는 변수인 경우, 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지는 경우,

및 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지지 않는 경우로 구분하여 토큰값들의 실행 중 일치 여부를 판단하는 것을 특징으로 한다.

<15> 이때, 상기 토큰값들의 실행 중 일치 여부는, 두 문장의 토큰이 모두 상수일 경우에는 해당 토큰 문자열의 동일 여부; 두 문장의 토큰 중 하나는 상수이고 하나는 변수인 경우에는 해당 변수를 상수로 치환하여 토큰 문자열의 동일 여부; 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지는 경우에는 두 문장 중에서 선행 문장으로부터 후행 문장에 이르는 제어흐름상에 해당 변수의 정의 존재 여부; 및 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지지 않는 경우에는 해당 변수를 원본 변수로 치환하여 두 문장 중에서 선행 문장으로부터 후행 문장에 이르는 제어흐름상에 해당 변수의 정의 존재 여부에 따라 판단하는 것이 바람직하다.

【발명의 구성 및 작용】

<16> 이하, 첨부된 도면을 참조하여 본 발명을 상세히 설명하기로 한다.

<17> 도 4 는 본 발명에 따라 악성 행위 패턴을 감지하기 위한 검사 대상 문장의 구분개념도로서, 검사 대상인 두 문장에 각각 포함된 토큰(변수 또는 상수)값들의 실행 중 일치 여부를 판단하여 악성 코드 패턴을 감지한다. 즉, 문장 Si에 포함된 변수 또는 상수 Ti와 문장 Sj에 포함된 변수 또는 상수 Tj가 실행시에 동일한 값을 가질 것인지를 실행 전의 코드를 통해 판단함으로써 악성 행위 패턴을 감지할 수 있다.

<18> 도 4 에 도시된 것처럼 스크립트에 존재하는 두 개의 변수 또는 상수가 동일한 값을 가지는가의 여부는 4 가지의 경우로 나누어 고찰해 볼 수 있다. 즉, 두 문장의 토큰

이 모두 상수일 경우(이하, '제 1 유형'이라 함), 두 문장의 토큰 중 하나는 상수이고 하나는 변수인 경우(이하, '제 2 유형'이라 함), 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지는 경우(이하, '제 3 유형'이라 함), 및 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지지 않는 경우(이하, '제 4 유형'이라 함)로 구분할 수 있다.

<19> 유형별로 살펴보면, 제 1 유형은 해당 토큰 문자열의 동일 여부로 토큰값들의 실행 중 일치 여부를 판단한다. 제 2 유형은 실행 전의 코드 분석을 통해서 토큰값들이 실행시에 동일한 값을 가질 것인가를 알 수 없다. 따라서, 상수 전파(constant propagation)를 수행하여 해당 변수가 상수로 치환될 경우에 토큰 문자열의 동일 여부로 토큰값들의 실행 중 일치 여부를 판단한다.

<20> 제 3 유형은 두 문장 중에서 선행 문장으로부터 후행 문장에 이르는 모든 제어흐름상에 해당 변수의 정의 존재 여부를 판단한다. 이때, 변수 정의가 존재하지 않는다면 항상 동일한 값을 가지는 것으로 판단한다. 제 4 유형은 해당 변수를 원본 변수로 치환하는 복사 전파(copy propagation)를 수행하여 두 문장 중에서 선행 문장으로부터 후행 문장에 이르는 모든 제어흐름상에 해당 변수의 정의 존재 여부를 판단한다. 이때, 변수 정의가 존재하지 않는다면 항상 동일한 값을 가지는 것으로 판단한다.

<21> 한편, 변수의 정의란 프로그램 언어 이론에서 지칭하는 바와 같이 해당 변수에 값을 대입하는 문장을 의미한다. 상수 전파는 자료 흐름 분석에 널리 사용되고 있는 기법으로, 프로그램 실행시 항상 특정 상수값을 가지게 될 변수 또는 수식을 찾아내고, 이러한 상수값을 가능한 프로그램 코드의 많은 부분으로 전파하는 것을 목적으로 한다. 복사

전파도 이와 유사한 기법으로, 'x = y' 형태의 복사문을 통해 실행시 항상 동일한 값을 가지게 될 변수를 찾아내어 치환함으로써 복사본의 수를 줄이는 기법을 지칭한다. 즉, 'x = y' 형태의 복사문 s가 있을 때, x의 사용 u가 y로 대치되고 복사문 s가 제거될 수 있으려면, 다음의 조건을 만족해야 한다.

<22> i) 문장 s가 u에 도달하는 x의 유일한 정의이어야 한다.

<23> ii) s로부터 u에 이르는 모든 경로에 대하여 y의 정의가 없어야 한다.

<24> 예컨대, 다음과 같은 프로그램 코드를 가정한다.

<25> x = y

<26> z = fso.getfile(x)

<27> 두번째 문장은 'z = fso.getfile(y)'로 바뀔 수 있으며, 이것이 변수 x의 유일한 사용이라면 첫번째 문장인 'x = y'는 삭제가 가능하게 된다. 이것은 각 기본 블럭마다 변수 복사에 관련된 사용과 정의를 구해내고, 제어 흐름 그래프를 따라 정보를 전파하는 반복적 알고리즘에 의해 구해질 수 있다. 기본 블럭이란 제어 흐름상에서 오직 하나의 진입점과 하나의 진출점만을 가지는 일련의 문장들을 의미하며, 제어 흐름 그래프의 노드(node)가 된다. 하나의 기본 블럭에서 다른 기본 블럭에 제어 흐름이 존재함은 제어 흐름 그래프의 에지(edge)로 표현된다. 도 5 는 이러한 제어 흐름 그래프의 실예를 나타낸다. 제어 흐름 그래프가 주어졌을 때, 하나의 기본 블럭에 도달하는 유효한 복사의 계산에 사용될 집합을 정의하면 다음과 같다.

<28> *in[B]* = 기본 블럭 B 의 선행 노드까지 유효한 복사

<29> *out[B]* = 기본 블럭 B 의 실행 직후까지 유효한 복사

<30> *c_gen[B]* = 기본 블럭 B 에서 정의된 복사

<31> *c_kill[B]* = 기본 블럭 B 에서 새롭게 정의되어 무효화된 복사

<32> 이때, *in[B]* 와 *out[B]* 는 다음의 수학식에 따라 계산될 수 있다.

<33>

【수학식 1】 $$in[B] = \bigcap_{P\ a\ predecessor\ of\ B} out[P],$$: B 가 프로그램 시작 블럭이 아닐 때

<34>

【수학식 2】 $$in[B_1] = \emptyset,$$: B 가 프로그램 시작일 때

<35>

【수학식 3】 $$out[B] = c\_gen[B] \bigcup (in[B] - c\_kill[B])$$

<36> 상수 및 복사 전파가 이루어지고 나면 해당 변수의 값이 동일할 것임을 알 수 있다. 즉, 두 토큰이 동일한 상수이거나, 모두 동일한 변수 이름과 범위를 가지며 이것이 위치한 두 문장사이에 해당 변수의 정의가 없어야 한다.

<37> 다른 한편, 문장 Si에 포함된 토큰 Ti와 문장 Sj에 포함된 토큰 Tj가 동일한 변수 또는 상수 V라 할 때, 스크립트에서 얻어진 제어 흐름 그래프 G를 이용하여 이를 검사하는 알고리즘은 다음과 같다. 즉, 제 1 단계는 워크-리스트 W : = { Si }, 두 값이 동일함을 알리는 플래그 IsEqual := true, 두 노드간에 제어 흐름이 존재함을 알리는 플래그

IsLinked := false 로 판단한다. 제 2 단계는 W로부터 한 노드를 꺼내어 c라 하는데, 꺼낼 노드가 없으면 제 5 단계로 간다.

<38> 이어서, 제 3 단계는 V가 변수이고 c가 변수 V의 정의이며, Sj에 이르는 경로가 존재하면 IsEqual := false, 그렇지 않고 c가 문장 Sj이면 IsLinked := true, 그렇지 않고 c가 프로그램 종료 노드나 이미 방문했던 노드가 아니면 링크된 모든 노드를 W에 넣는다. 제 4 단계는 검사를 계속하기 위해 제 2 단계로 간다. 마지막으로, 제 5 단계는 sLinked = true이고 IsEqual = true이면 변수 또는 상수 V는 Si와 Sj에서 동일한 값을 가진 것으로 판단한다.

<39> 상술한 알고리즘의 제 3 단계에서 'c가 변수 V의 정의' 인 것은 다음 중 하나를 의미한다.

<40> i) c가 대입문이고 변수 V가 좌변에 있을 때

<41> ii) c가 함수 또는 프로시져를 호출하는 문장이고, 변수 V가 해당 함수 호출 p 의 MAY_DEF(p) 집합에 속해 있을 때

<42> 여기에서 MAY_DEF(p) 집합은 프로시져 p에서 정의될 수 있는 전역 변수와 참조 인자의 집합이다. 이것은 프로시져 내에서 정의된 전역변수, 호출시에 참조 형태로 주어진 실인자, 그리고 해당 프로시져가 호출하는 프로시져의 MAY_DEF 집합의 합집합으로 구해질 수 있으므로, 계산이 간단하다. 즉, 각 프로시져 내부에서 정의된 전역변수와 호출 그래프만으로 대부분의 계산이 미리 이루어질 수 있다. 따라서, 두 문장 사이의 경로를 탐색하는 도중 프로시져 호출이 나타나면 해당 프로시져 내부까지 탐색하지 않고 MAY_DEF 집합만을 참조하면 분석 시간을 크게 단축시킬 수 있다.

<43> 결론적으로, 상기와 같은 검사를 수행하기 위해서는 도 6 과 같은 절차가 필요하다. 도면을 참조하면, 주어진 스크립트를 읽어 제어 흐름 그래프를 생성한다(S610). 이어서, 생성된 제어 흐름 그래프에서 자료 흐름 분석을 통해 복사/상수 전파를 수행한다(S620). 이때, 복사/상수 전파의 결과물은 변형된 스크립트이거나, 변형된 스크립트에 대응되는 새로운 제어 흐름 그래프가 된다. 마지막으로, 악성 행위 패턴 감지를 실시한다(S630). 이때, 두 문장에 존재하는 두 개의 토큰값이 실행 중 일치할 것인가를 판단할 때에는 상기 S620 단계에서 얻어진 제어 흐름 그래프를 이용하여 상술한 알고리즘을 이용한다.

【발명의 효과】

<44> 이상 설명한 바와 같이 제어흐름과 자료흐름을 고려한 악성 행위 패턴 감지 방법은 종래의 변수명 비교에서 발생할 수 있는 긍정 오류를 배제하고 부정 오류를 낮추어 악성 행위 감지의 정확도를 향상시킬 수 있다.

【특허청구범위】

【청구항 1】

검사 대상인 두 문장에 각각 포함된 토큰(변수 또는 상수)값들의 실행 중 일치 여부를 판단하여 악성 코드 패턴을 감지하되,

두 문장의 토큰이 모두 상수일 경우와, 두 문장의 토큰 중 하나는 상수이고 하나는 변수인 경우, 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지는 경우, 및 두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지지 않는 경우로 구분하여 토큰값들의 실행 중 일치 여부를 판단하는 것을 특징으로 하는 제어흐름과 자료흐름을 고려한 악성 행위 패턴 감지 방법.

【청구항 2】

제 1 항에 있어서 상기 토큰값들의 실행 중 일치 여부는,

두 문장의 토큰이 모두 상수일 경우에는 해당 토큰 문자열의 동일 여부;

두 문장의 토큰 중 하나는 상수이고 하나는 변수인 경우에는 해당 변수를 상수로 치환하여 토큰 문자열의 동일 여부;

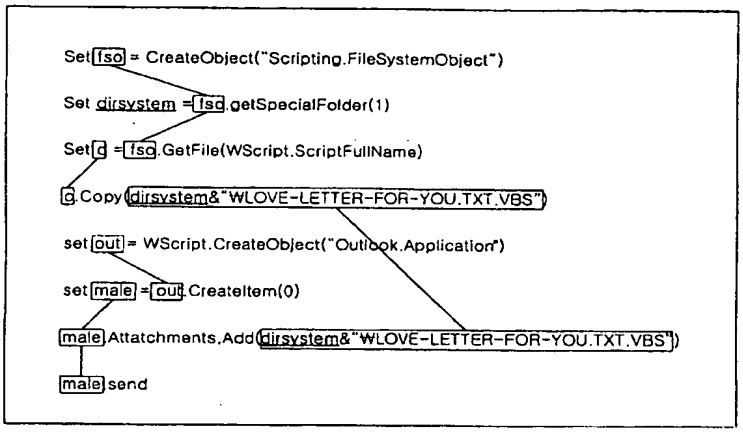
두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지는 경우에는 두 문장 중에서 선행 문장으로부터 후행 문장에 이르는 제어흐름상에 해당 변수의 정의 존재 여부; 및

두 문장의 토큰이 모두 변수이면서 동일한 이름과 범위를 가지지 않는 경우에는 해당 변수를 원본 변수로 치환하여 두 문장 중에서 선행 문장으로부터 후행 문장에 이르는

제어흐름상에 해당 변수의 정의 존재 여부에 따라 판단하는 것을 특징으로 하는 제어흐름과 자료흐름을 고려한 악성 행위 패턴 감지 방법.

【도면】

【도 1】



【도 2】

```
set c = fso.GetFile(WScript.ScriptFullName)
...                                          // 변수 c 를 이용한 작업 완료
set c = fso.GetFile( test.txt )              // 새로운 c 값 정의
c.Copy( LOVE-LETTER-FOR-YOU.TXT.VBS )
```

【도 3】

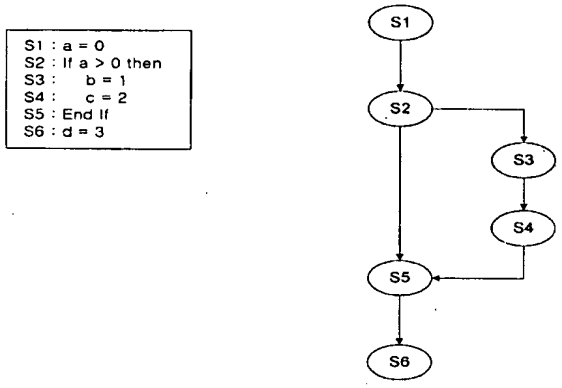
```
set c = fso.GetFile(WScript.ScriptFullName)
d = c
d.Copy( LOVE-LETTER-FOR-YOU.TXT.VBS )
```

【도 4】

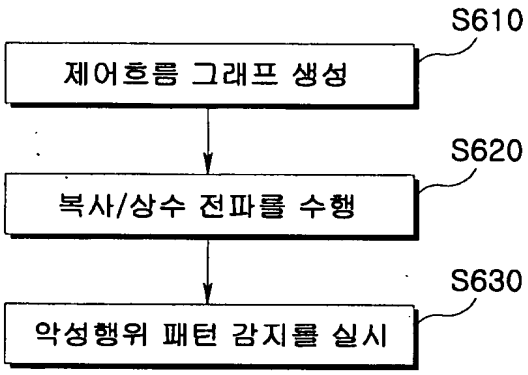
검사대상 문장의 구분

- 두 문장의 토큰이 모두 상수인 경우 : 제 1유형
- 두 문장의 토큰중 하나는 상수이고 하나는 변수인 경우 : 제 2유형
- 두 문장의 토큰이 모두 변수이면서 동일 변수인 경우 : 제 3유형
- 두 문장의 토큰이 모두 변수이면서 상이 변수인 경우 : 제 4유형

【도 5】


S1 : a = 0
S2 : If a > 0 then
S3 : b = 1
S4 : c = 2
S5 : End If
S6 : d = 3
S1
S2
S3
S4
S5
S6

【도 6】


S610
제어흐름 그래프 생성
S620
복사/상수 전파를 수행
S630
악성행위 패턴 감지를 실시